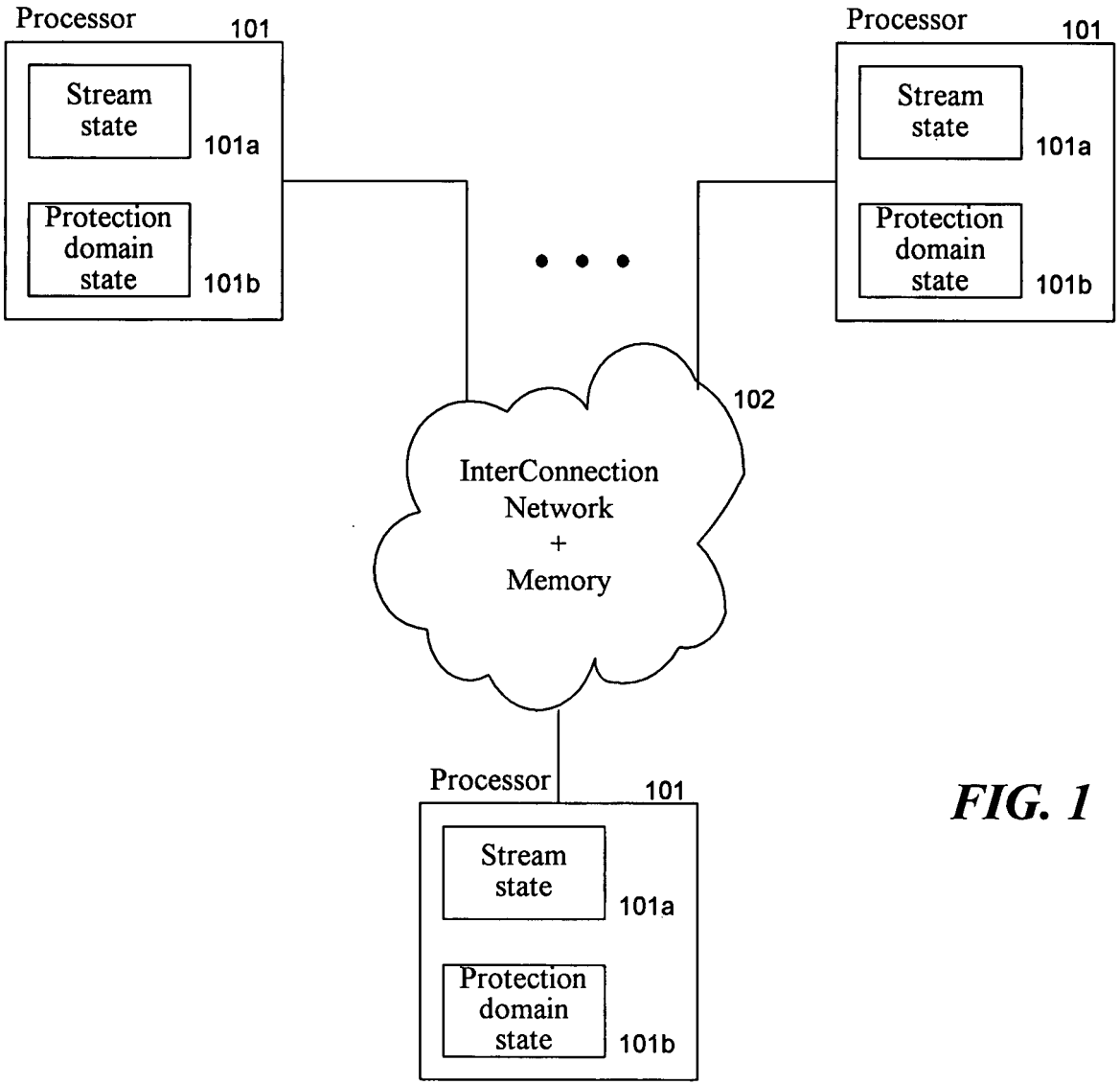
Processor
101
Stream state
101a
Protection domain state
101b
Processor
101
Stream state
101a
Protection domain state
101b
102
InterConnection Network + Memory
Processor
101
Stream state
101a
Protection domain state
101b

FIG. 1

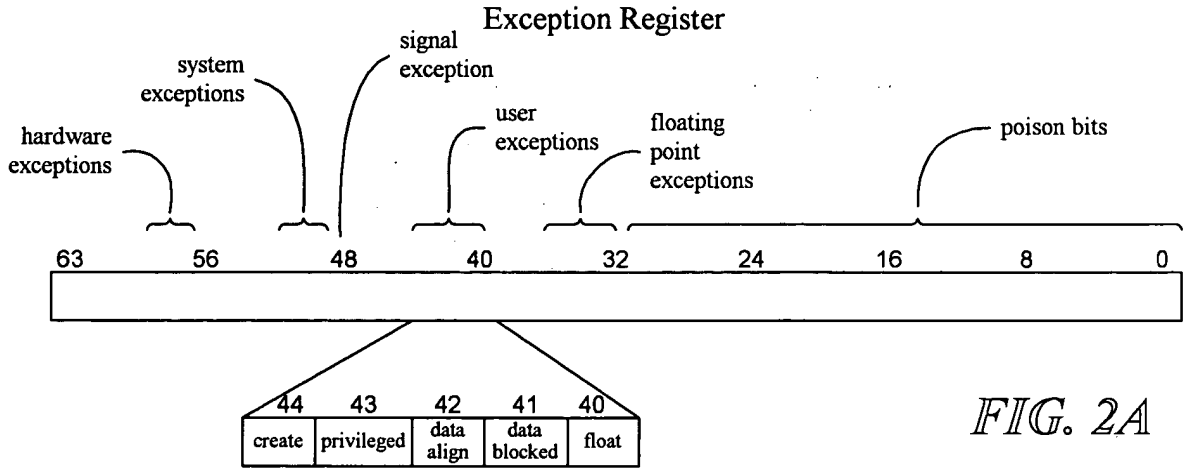

FIG. 2A

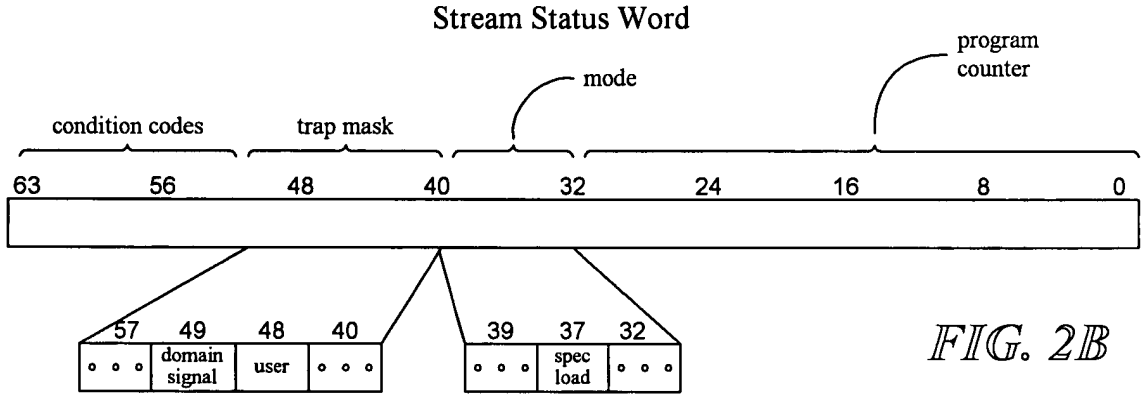

FIG. 2B

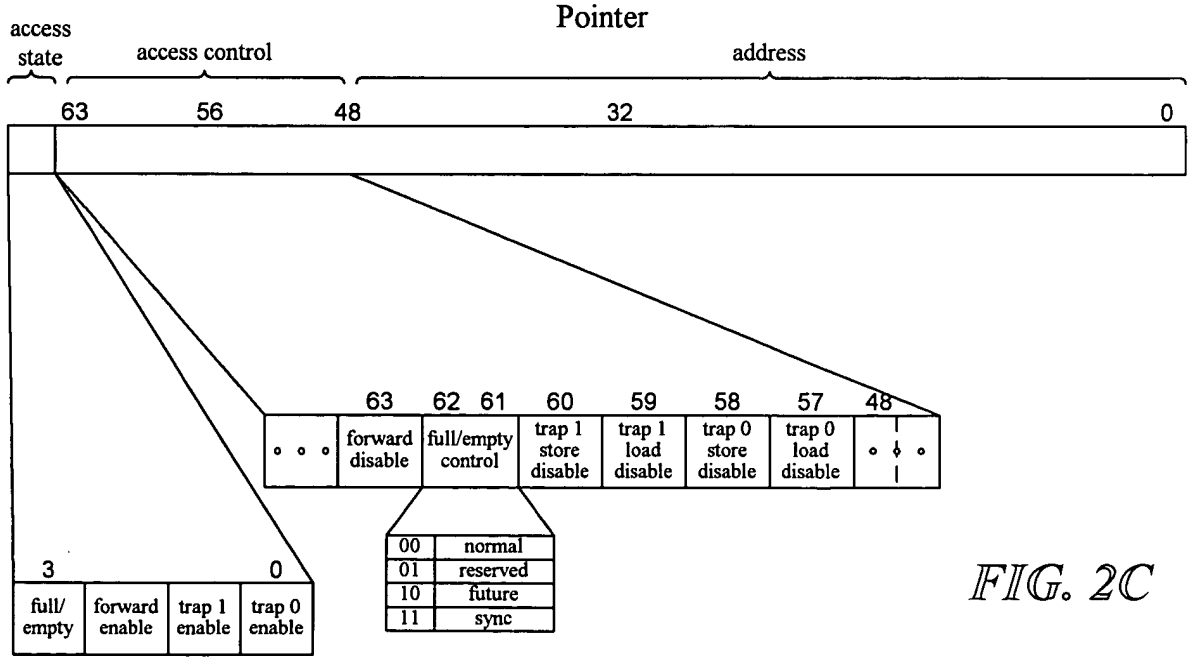

FIG. 2C

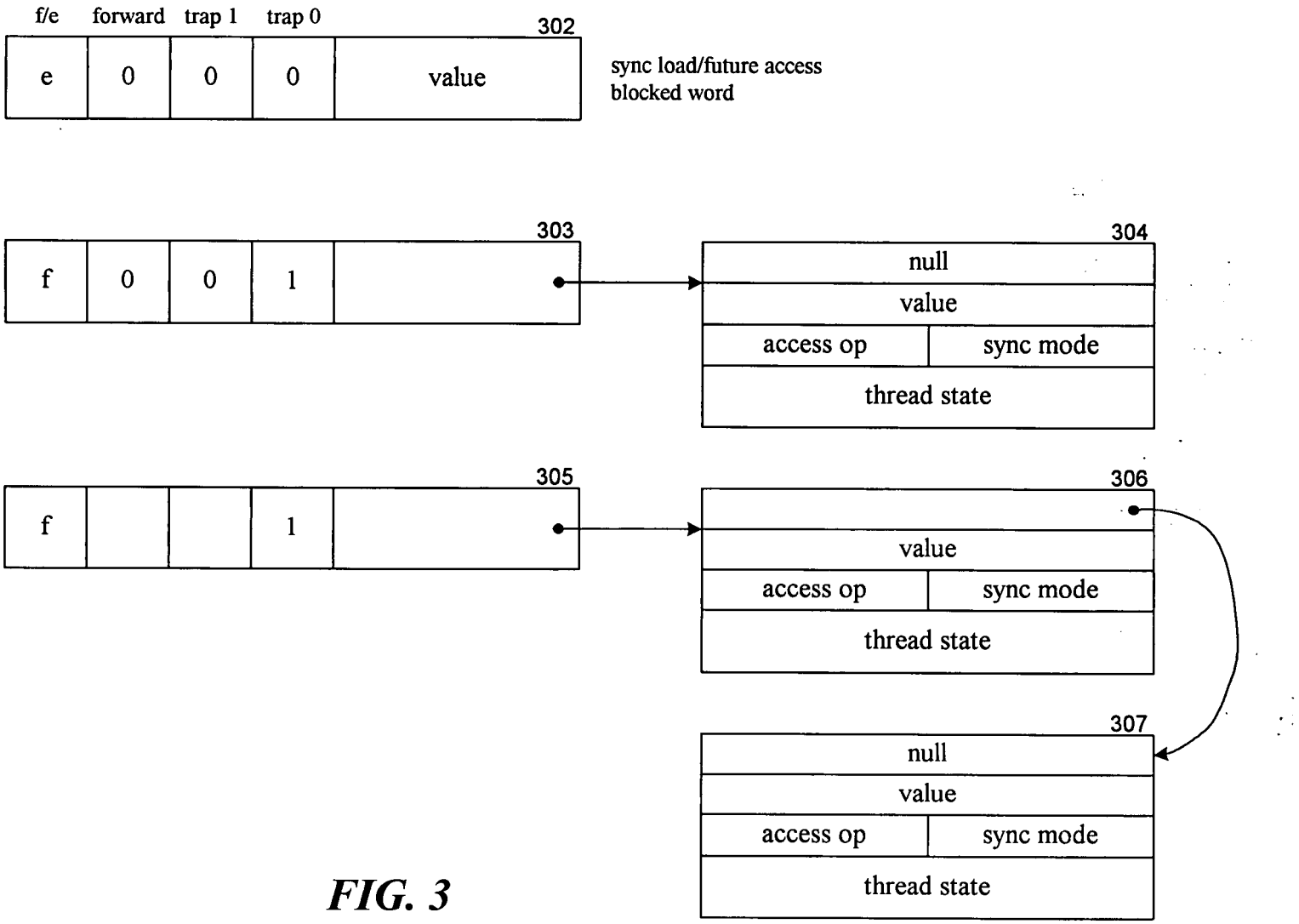
f/e
forward
trap 1
trap 0
302
e
0
0
0
value
sync load/future access
blocked word
303
f
0
0
1
304
null
value
access op
sync mode
thread state
305
f
1
306
value
access op
sync mode
thread state
307
null
value
access op
sync mode
thread state

FIG. 3

current f/e \ sync mode	load			store		
	normal	sync	future	normal	sync	future
full	succeed	succeed	succeed	succeed	block	succeed
	full	empty *	full	full	N/A	full
empty	succeed	block	block	succeed	succeed	block
	empty	N/A	N/A	full *	full *	N/A

401

FIG. 4

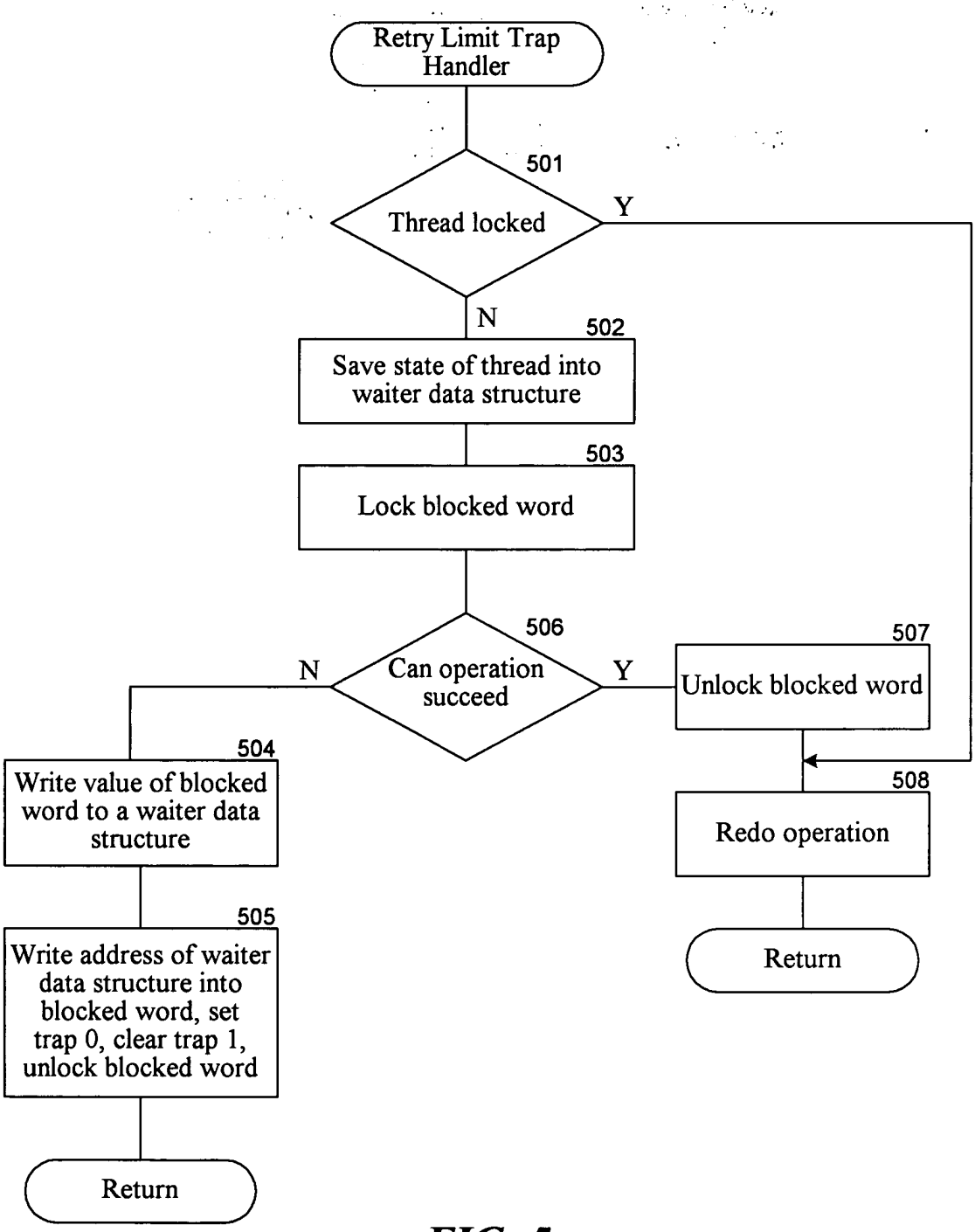
Retry Limit Trap Handler
501
Thread locked
Y
N
502
Save state of thread into waiter data structure
503
Lock blocked word
506
Can operation succeed
N
Y
507
Unlock blocked word
504
Write value of blocked word to a waiter data structure
508
Redo operation
505
Write address of waiter data structure into blocked word, set trap 0, clear trap 1, unlock blocked word
Return
Return

FIG. 5

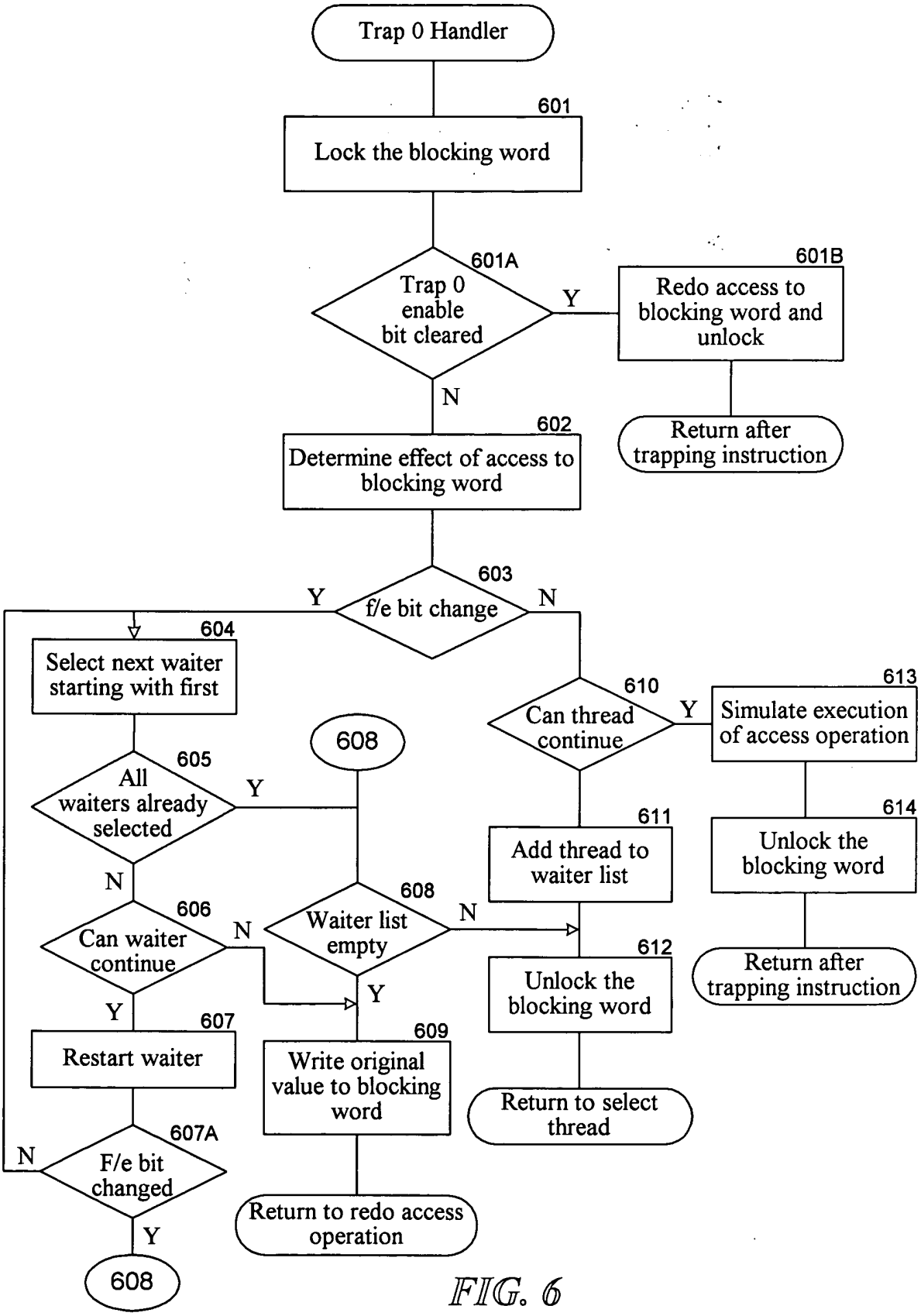
Trap 0 Handler
601
Lock the blocking word
601A
Trap 0 enable bit cleared
Y
601B
Redo access to blocking word and unlock
Return after trapping instruction
N
602
Determine effect of access to blocking word
603
f/e bit change
Y
N
604
Select next waiter starting with first
605
All waiters already selected
Y
N
606
Can waiter continue
N
Y
607
Restart waiter
607A
F/e bit changed
N
Y
608
608
Waiter list empty
608
N
Y
609
Write original value to blocking word
Return to redo access operation
610
Can thread continue
Y
611
Add thread to waiter list
612
Unlock the blocking word
Return to select thread
613
Simulate execution of access operation
614
Unlock the blocking word
Return after trapping instruction

FIG. 6

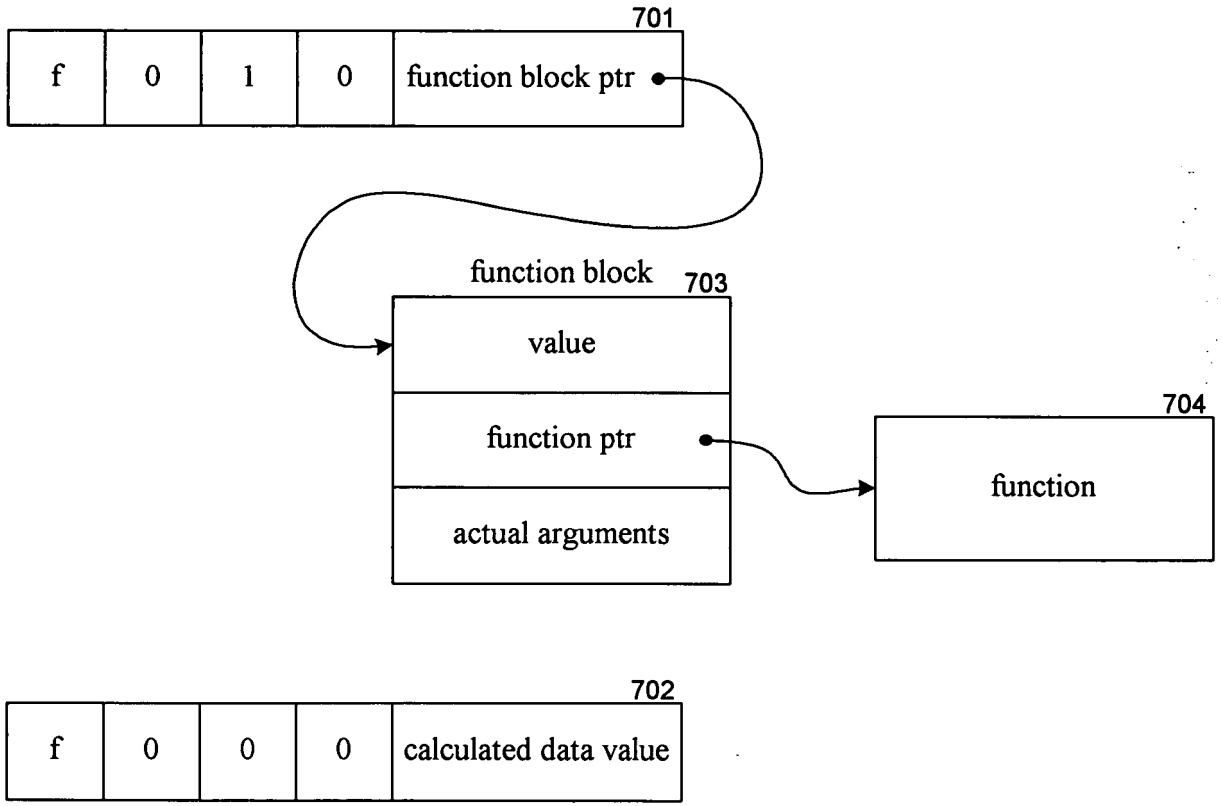
701
f
0
1
0
function block ptr
function block
703
value
function ptr
actual arguments
704
function
702
f
0
0
0
calculated data value

FIG. 7

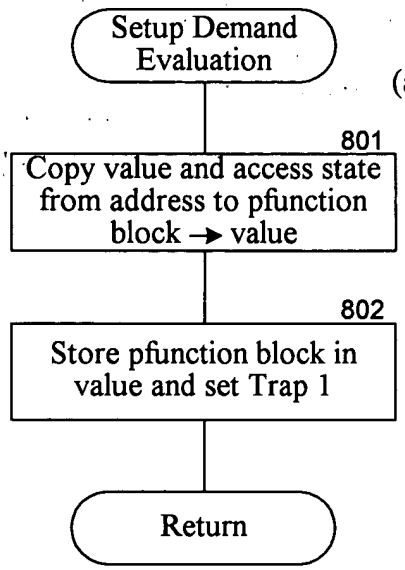
Setup Demand Evaluation
(address, pfunction block)
801
Copy value and access state from address to pfunction block → value
802
Store pfunction block in value and set Trap 1
Return

FIG. 8

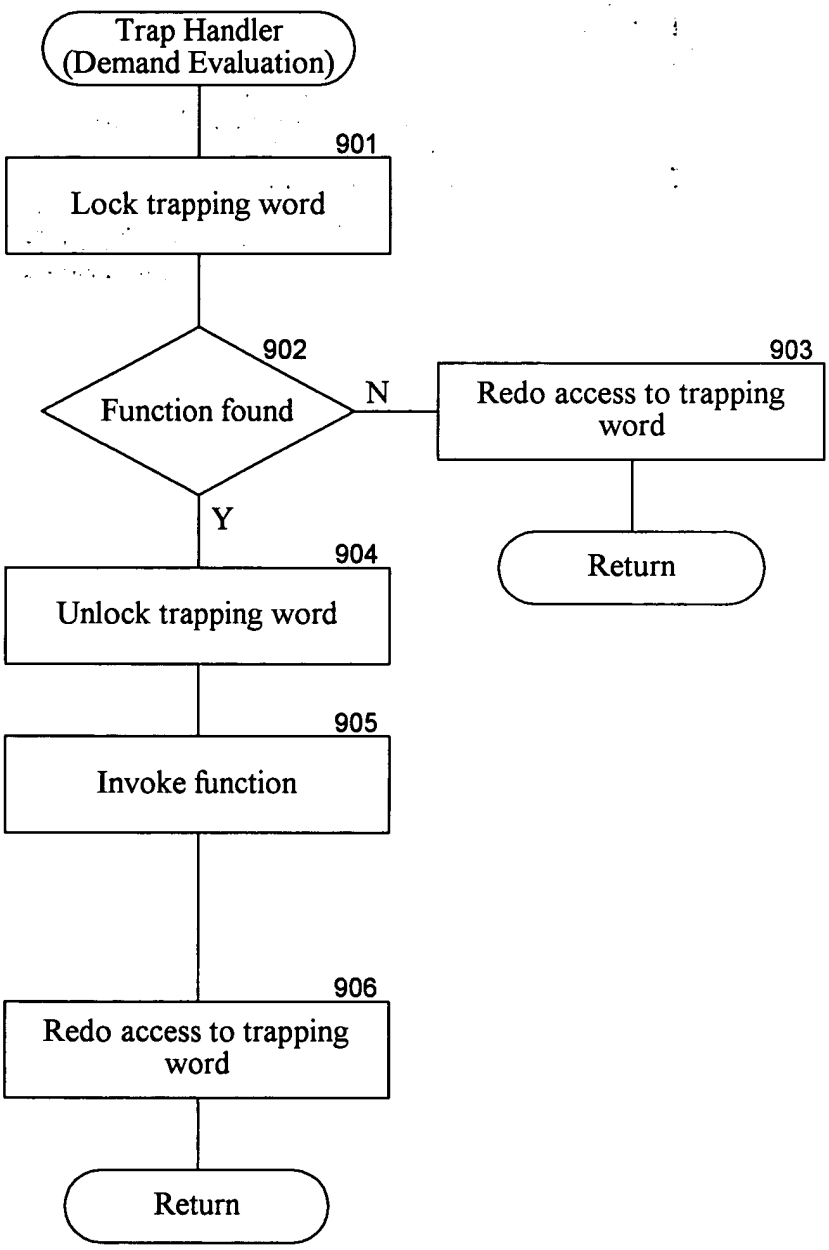
Trap Handler
(Demand Evaluation)
901
Lock trapping word
902
Function found
N
903
Redo access to trapping word
Return
Y
904
Unlock trapping word
905
Invoke function
906
Redo access to trapping word
Return

FIG. 9

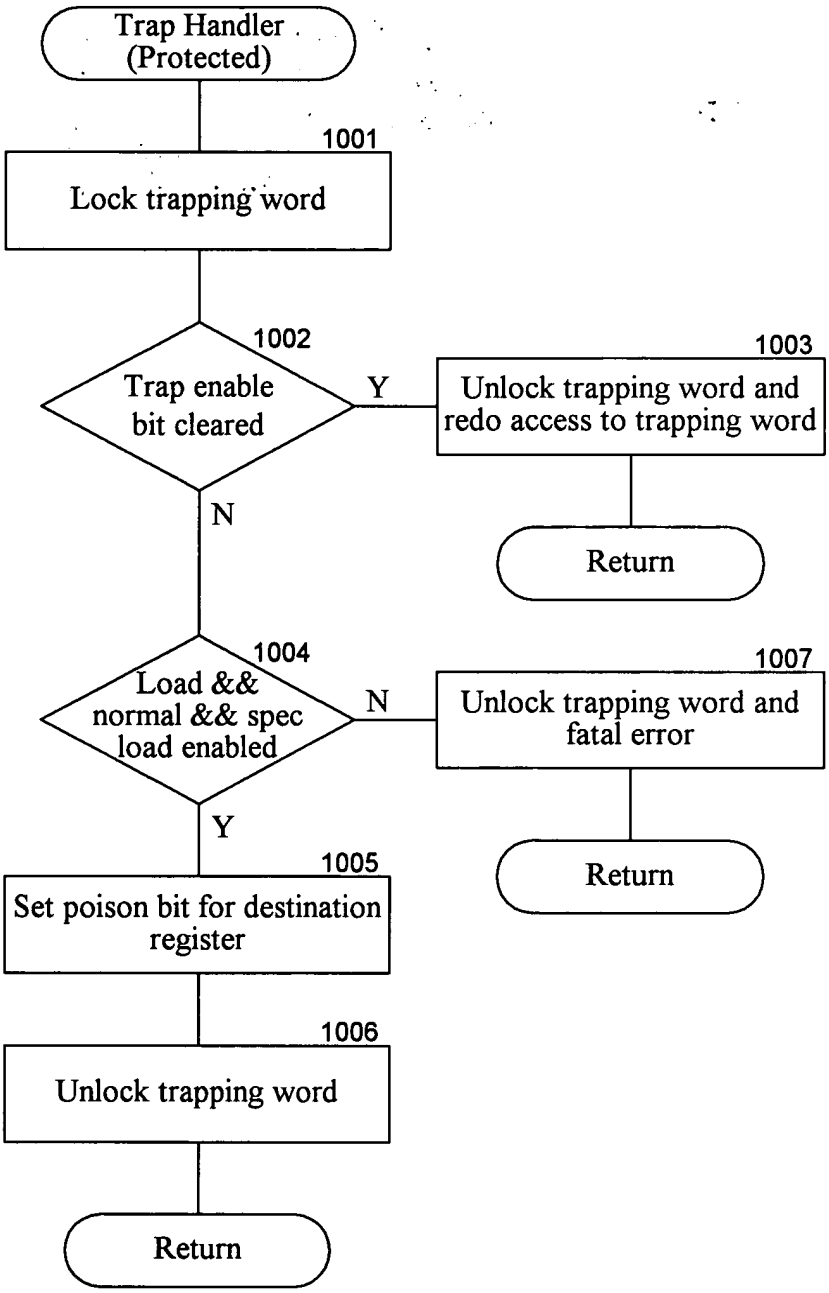
Trap Handler (Protected)
1001
Lock trapping word
1002
Trap enable bit cleared
Y
1003
Unlock trapping word and redo access to trapping word
Return
N
1004
Load && normal && spec load enabled
N
1007
Unlock trapping word and fatal error
Return
Y
1005
Set poison bit for destination register
1006
Unlock trapping word
Return

FIG. 10

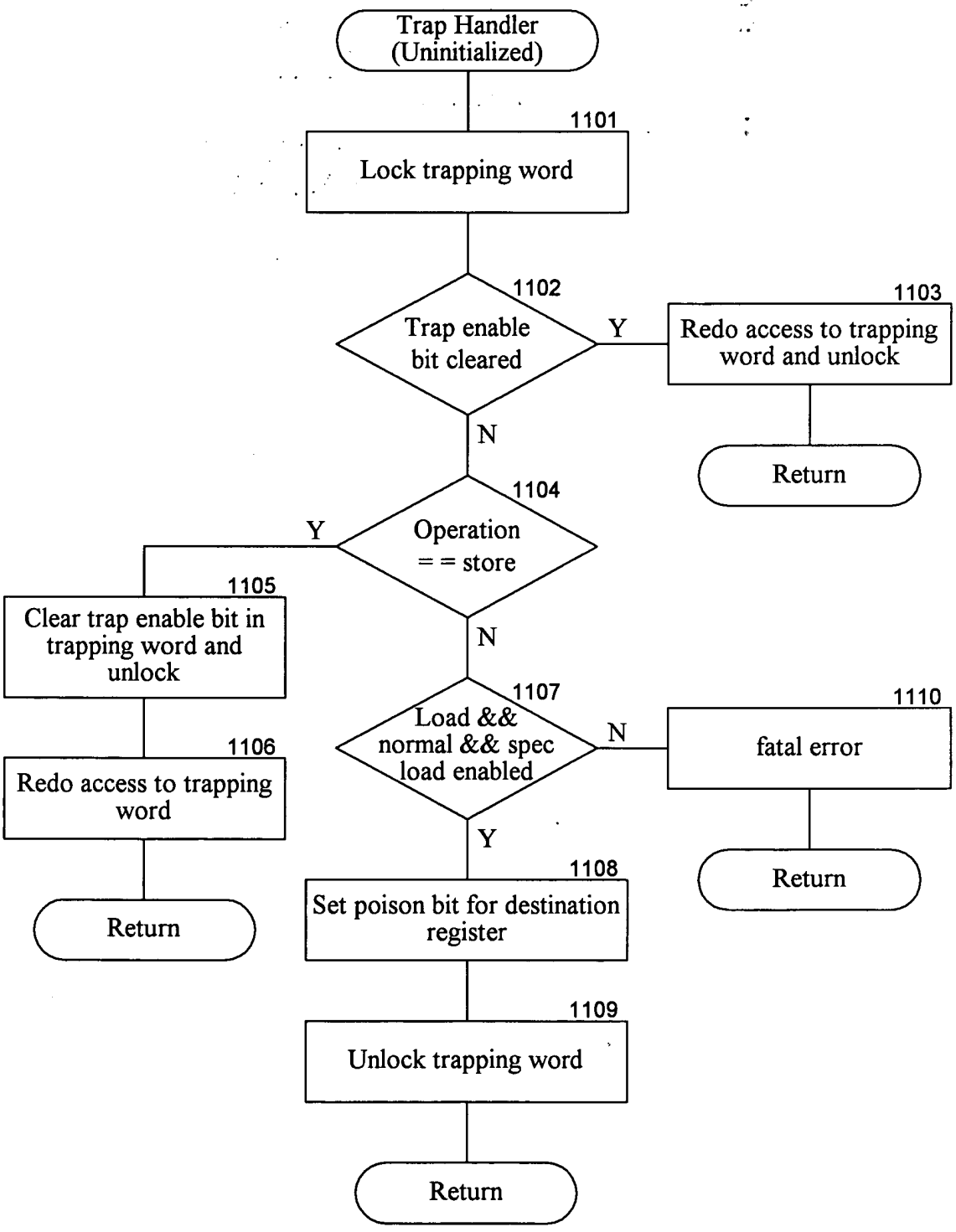
Trap Handler (Uninitialized)
1101
Lock trapping word
1102
Trap enable bit cleared
Y
1103
Redo access to trapping word and unlock
Return
N
1104
Operation == store
Y
1105
Clear trap enable bit in trapping word and unlock
1106
Redo access to trapping word
Return
N
1107
Load && normal && spec load enabled
N
1110
fatal error
Return
Y
1108
Set poison bit for destination register
1109
Unlock trapping word
Return

FIG. 11

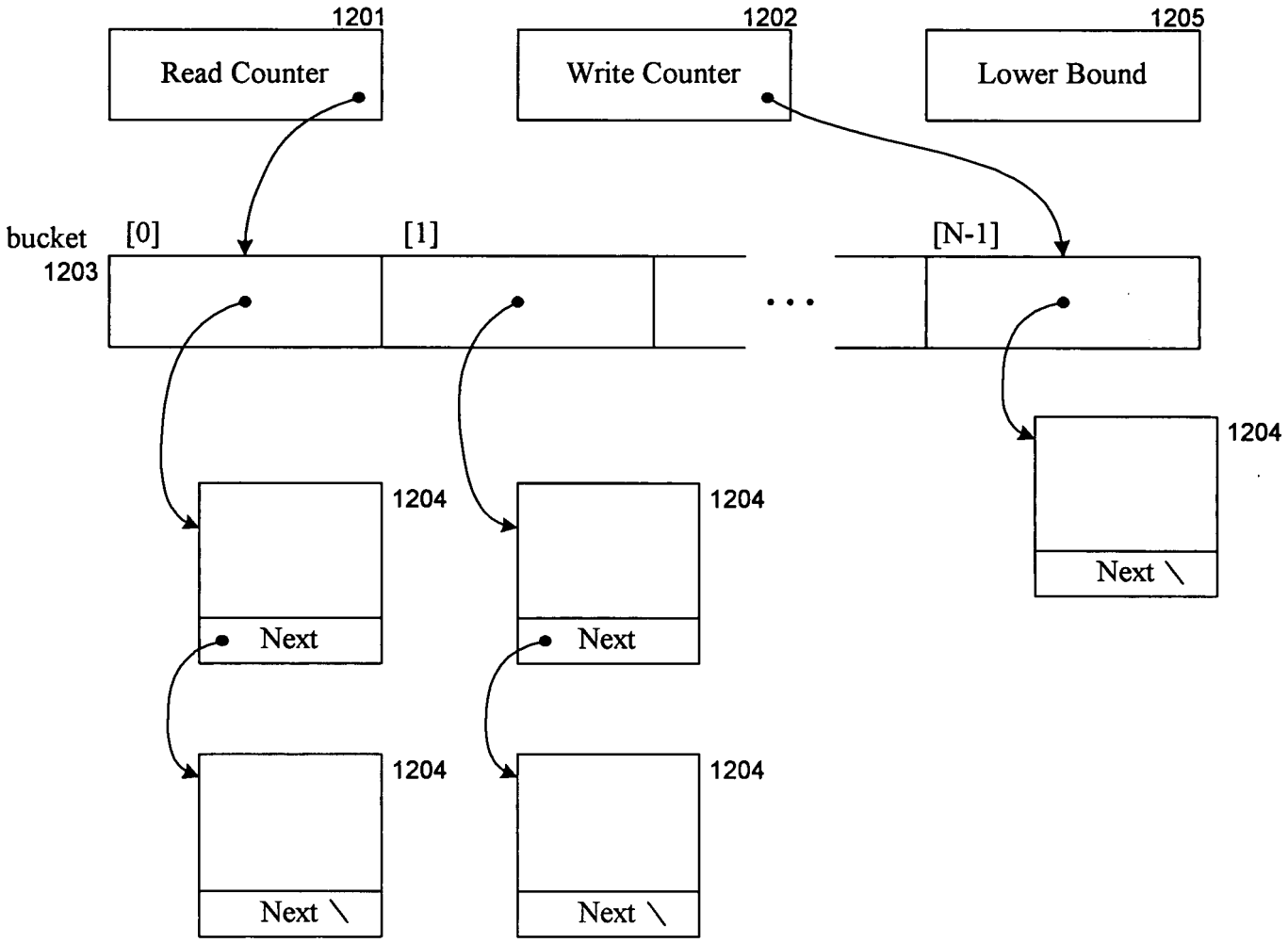
1201
Read Counter
1202
Write Counter
1205
Lower Bound
bucket
1203
[0]
[1]
[N-1]
1204
1204
1204
Next
Next
Next
1204
1204
Next
Next

FIG. 12

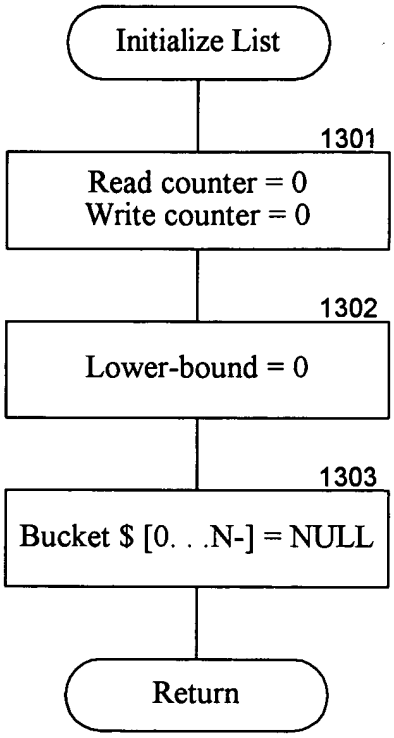
Initialize List
1301
Read counter = 0
Write counter = 0
1302
Lower-bound = 0
1303
Bucket $ [0. . .N-] = NULL
Return

FIG. 13

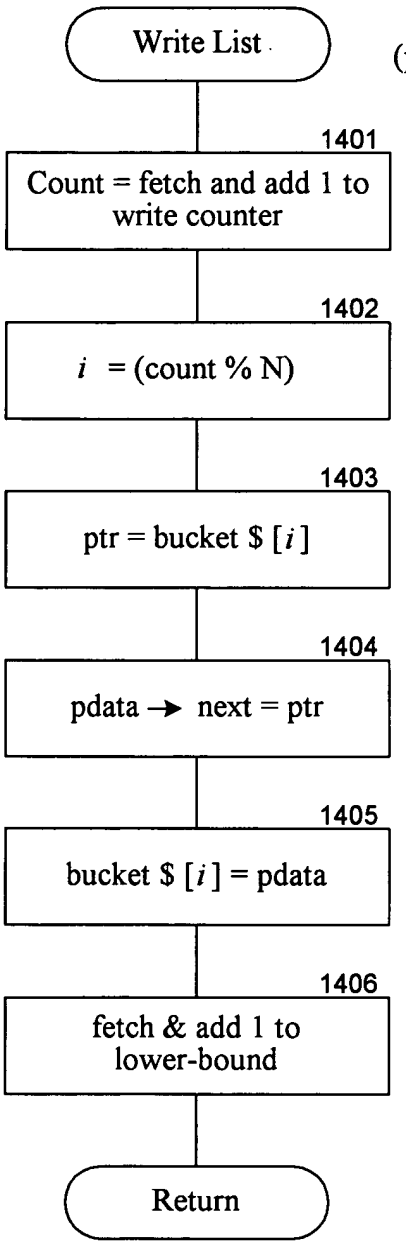
Write List
(pdata)
1401
Count = fetch and add 1 to write counter
1402
i = (count % N)
1403
ptr = bucket $ [i]
1404
pdata → next = ptr
1405
bucket $ [i] = pdata
1406
fetch & add 1 to lower-bound
Return

FIG. 14

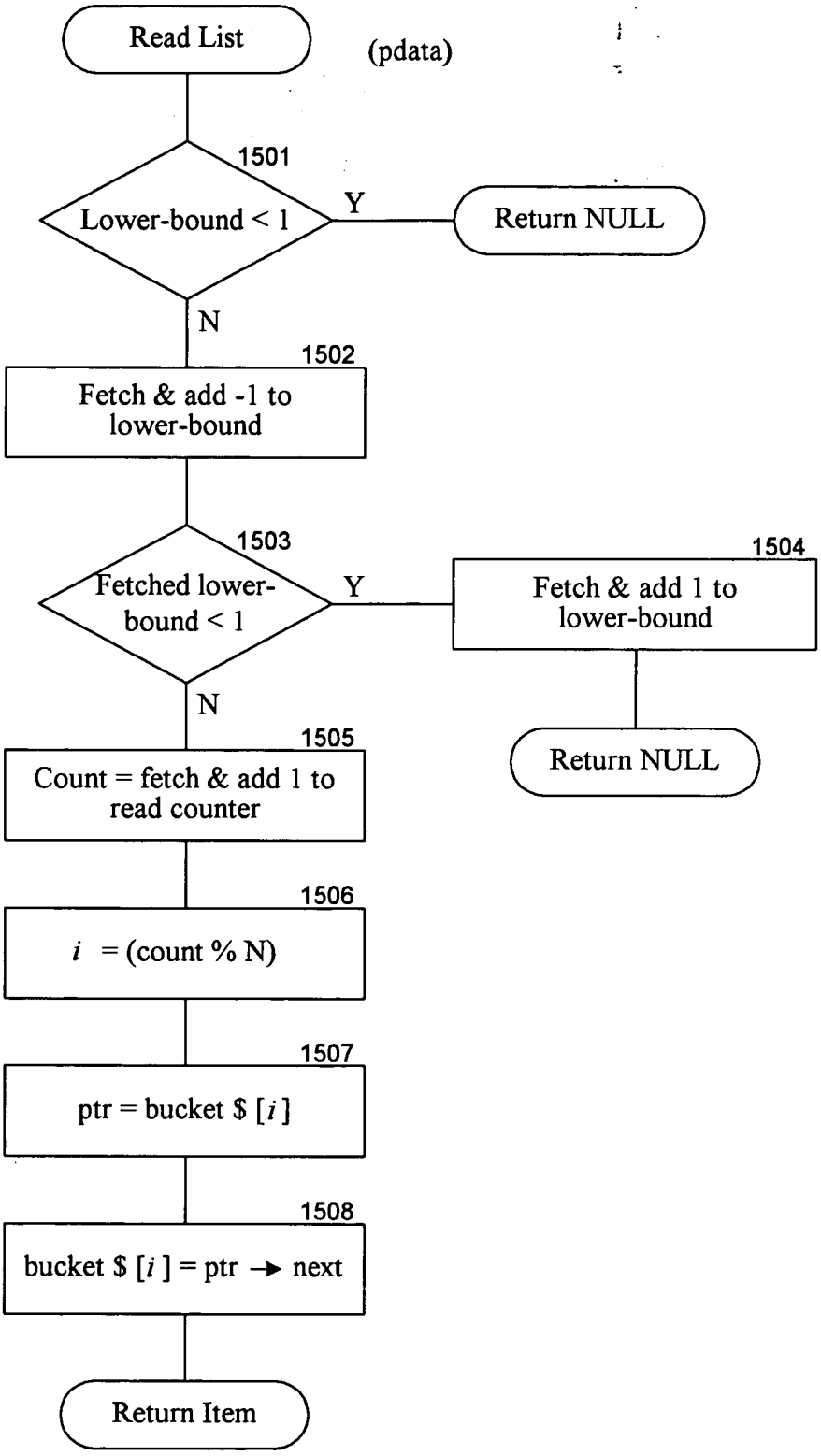
Read List
(pdata)
1501
Lower-bound < 1
Y
Return NULL
N
1502
Fetch & add -1 to lower-bound
1503
Fetched lower-bound < 1
Y
1504
Fetch & add 1 to lower-bound
Return NULL
N
1505
Count = fetch & add 1 to read counter
1506
i = (count % N)
1507
ptr = bucket $ [i]
1508
bucket $ [i] = ptr → next
Return Item

FIG. 15

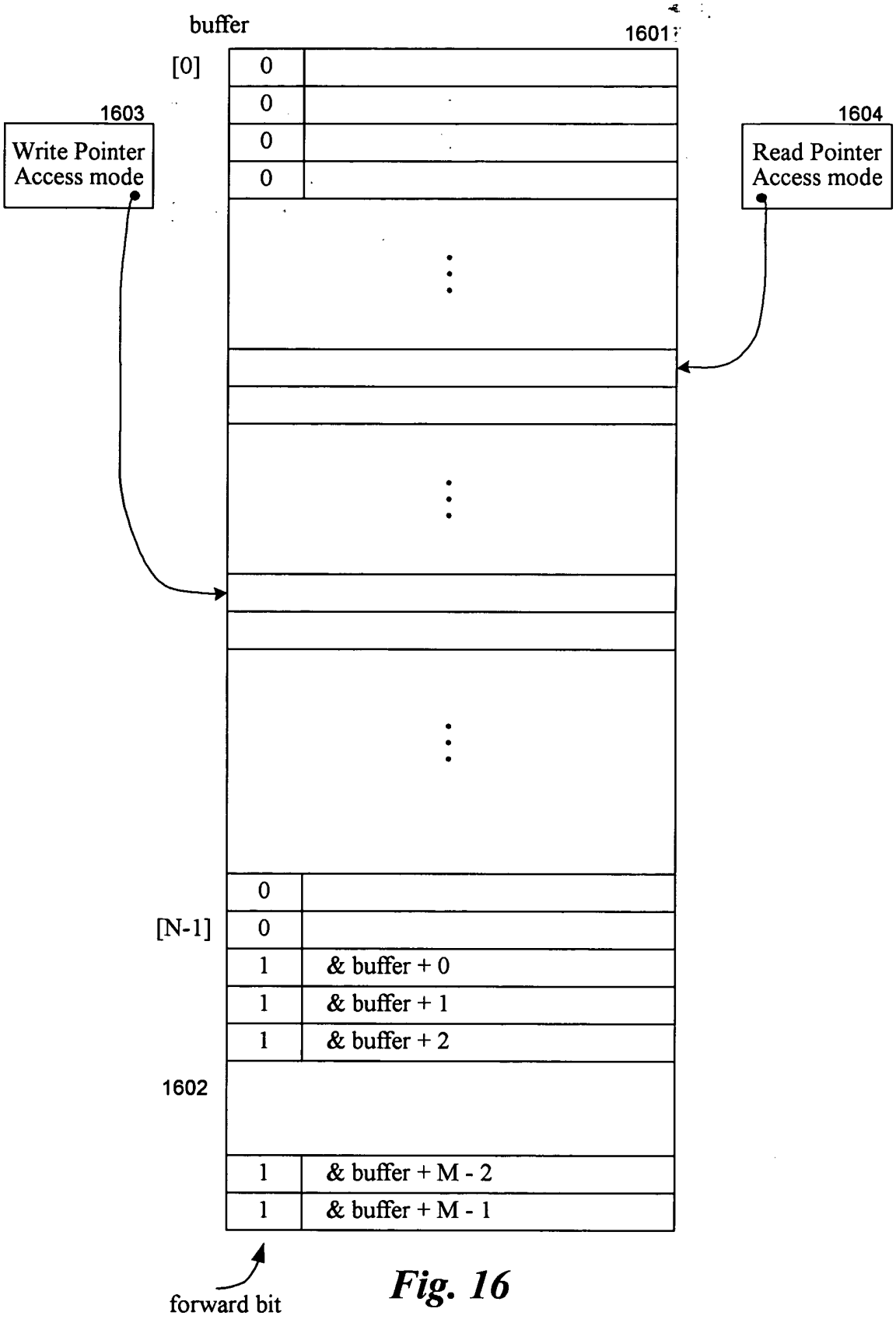
buffer
1601
[0]
0
0
0
0
1603
Write Pointer
Access mode
1604
Read Pointer
Access mode
[N-1]
0
0
1
& buffer + 0
1
& buffer + 1
1
& buffer + 2
1602
1
& buffer + M - 2
1
& buffer + M - 1
forward bit

Fig. 16

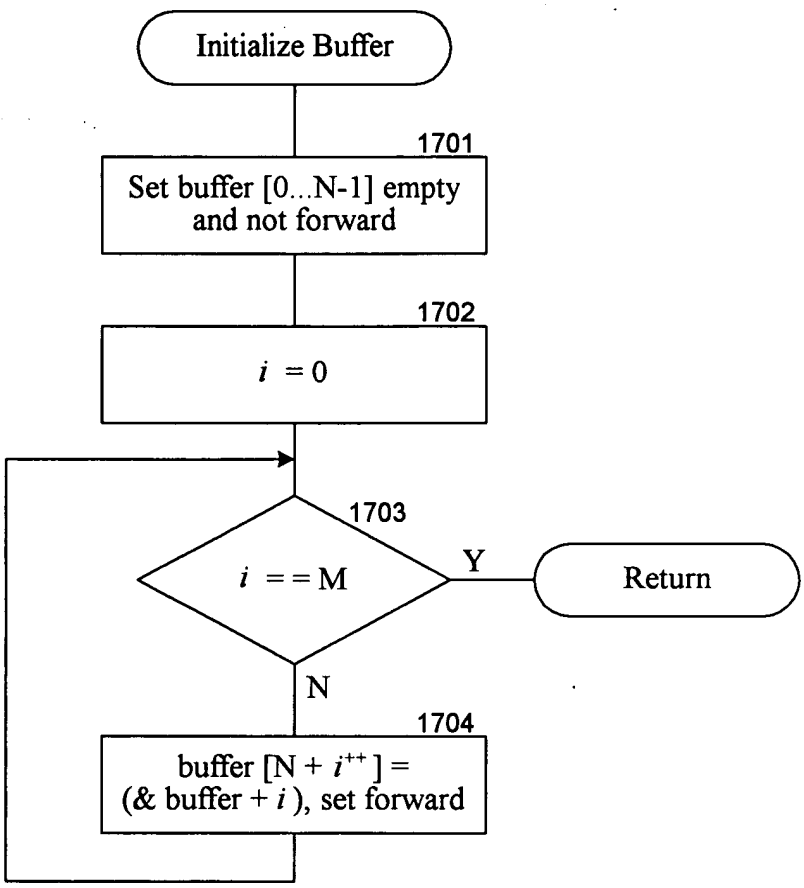
Initialize Buffer
1701
Set buffer [0...N-1] empty and not forward
1702
i = 0
1703
i == M
Y
Return
N
1704
buffer [N + i++] = (& buffer + i), set forward

FIG. 17

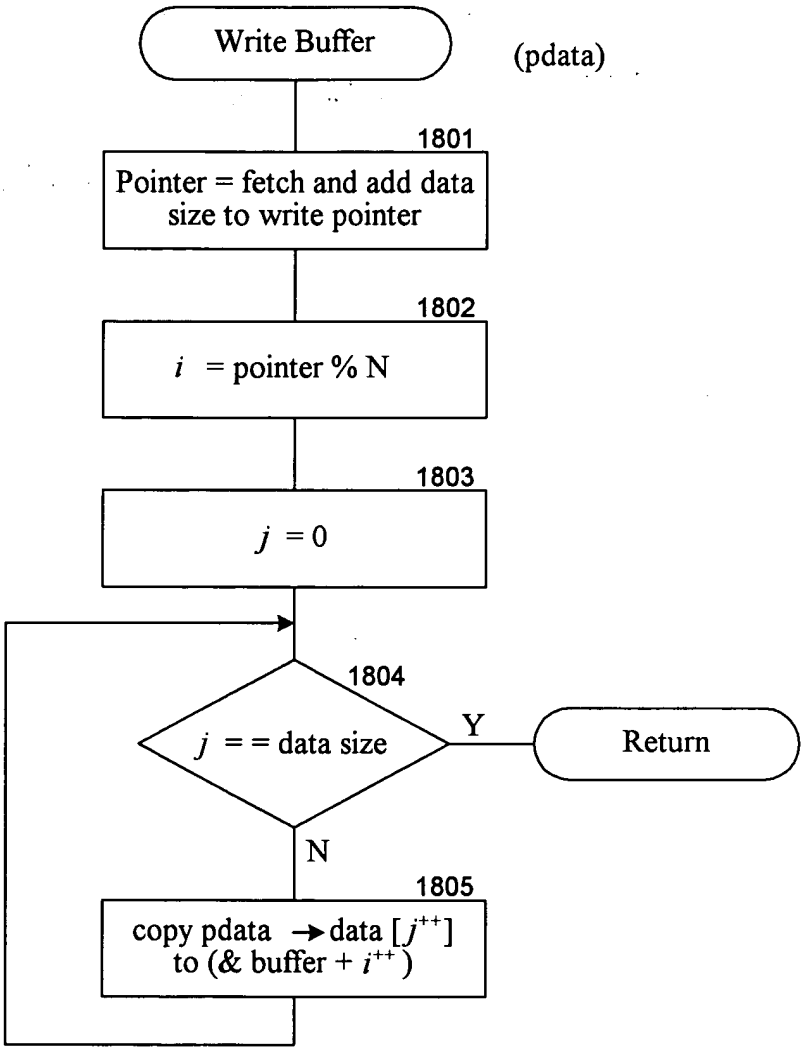
Write Buffer
(pdata)
1801
Pointer = fetch and add data size to write pointer
1802
i = pointer % N
1803
j = 0
1804
j = = data size
Y
Return
N
1805
copy pdata → data [j++] to (& buffer + i++)

FIG. 18

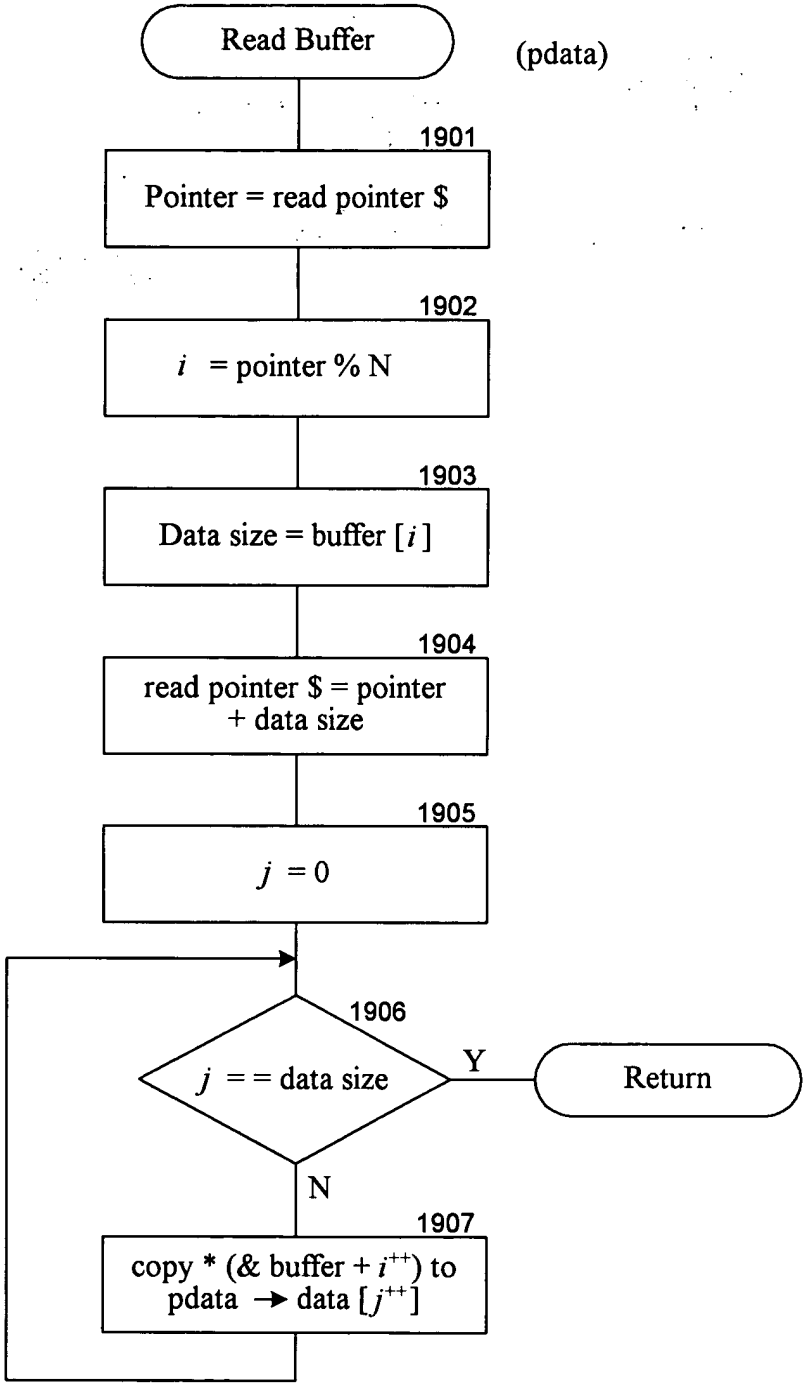
Read Buffer
(pdata)
1901
Pointer = read pointer $
1902
i = pointer % N
1903
Data size = buffer [i]
1904
read pointer $ = pointer + data size
1905
j = 0
1906
j = = data size
Y
Return
N
1907
copy * (& buffer + i++) to pdata → data [j++]

FIG. 19

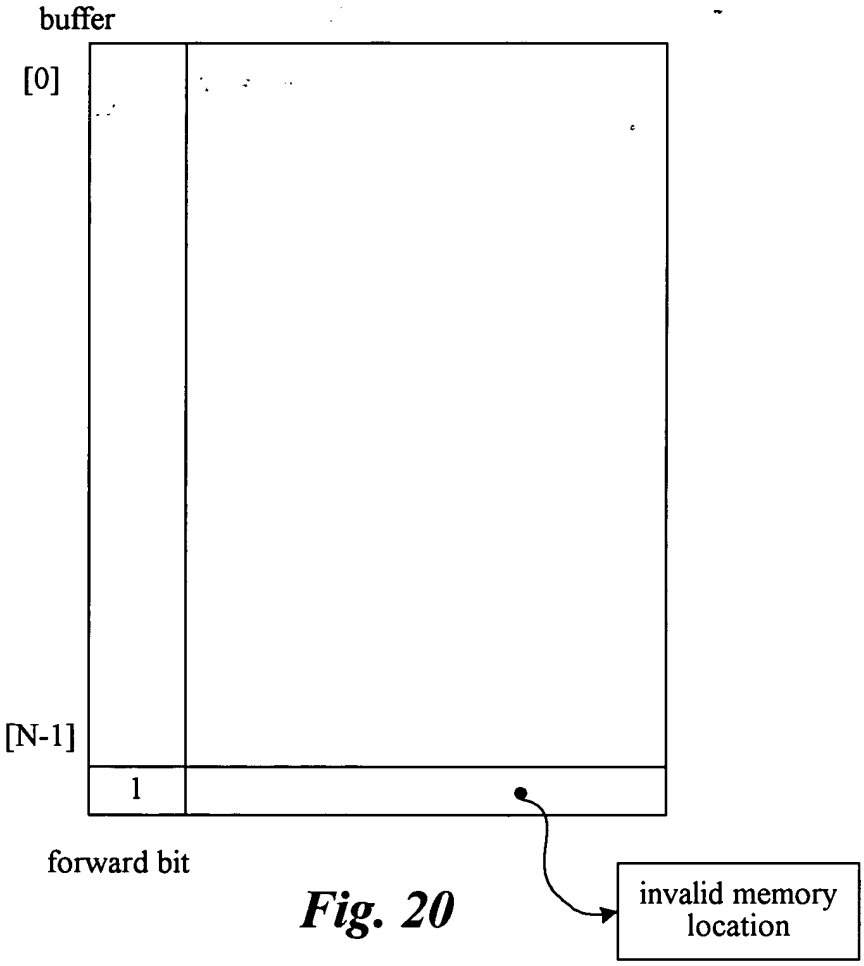

Fig. 20

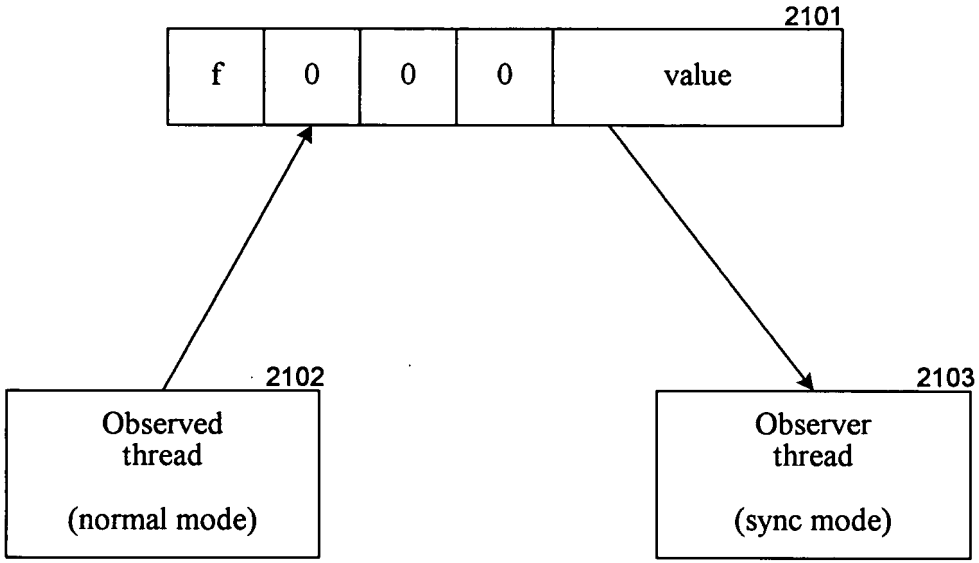
2101
f
0
0
0
value
2102
Observed thread
(normal mode)
2103
Observer thread
(sync mode)

FIG. 21

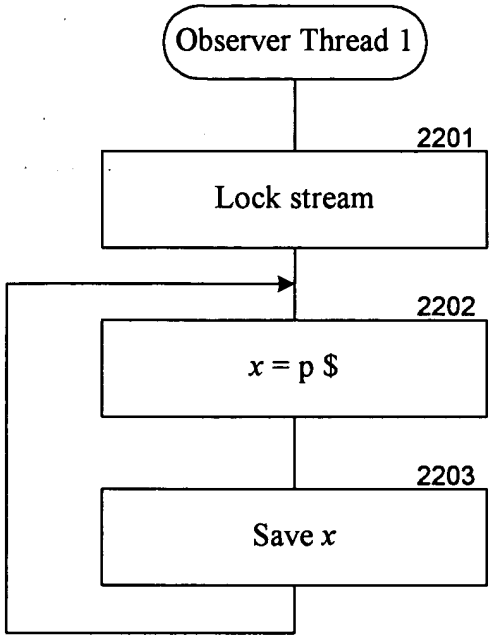

FIG. 22

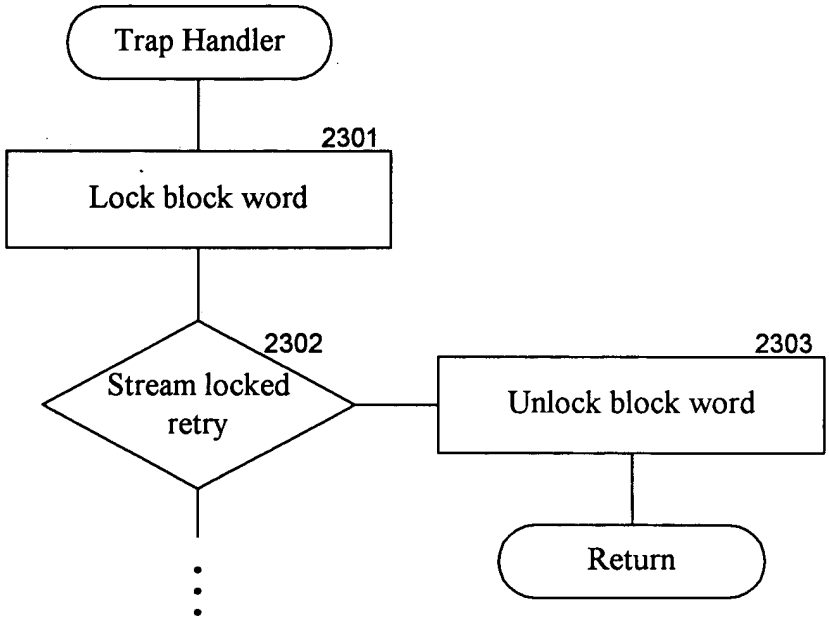
Trap Handler
2301
Lock block word
2302
Stream locked retry
2303
Unlock block word
Return

FIG. 23

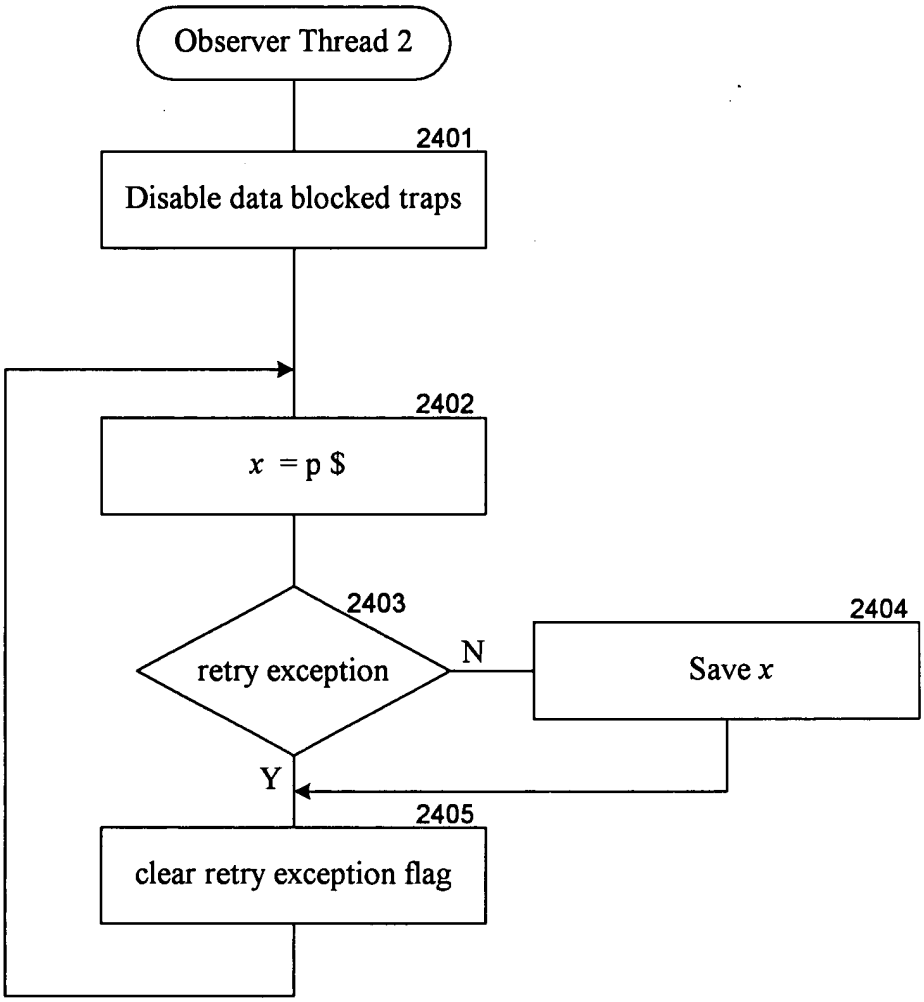
Observer Thread 2
2401
Disable data blocked traps
2402
x = p $
2403
retry exception
N
2404
Save x
Y
2405
clear retry exception flag

FIG. 24

Data Structure

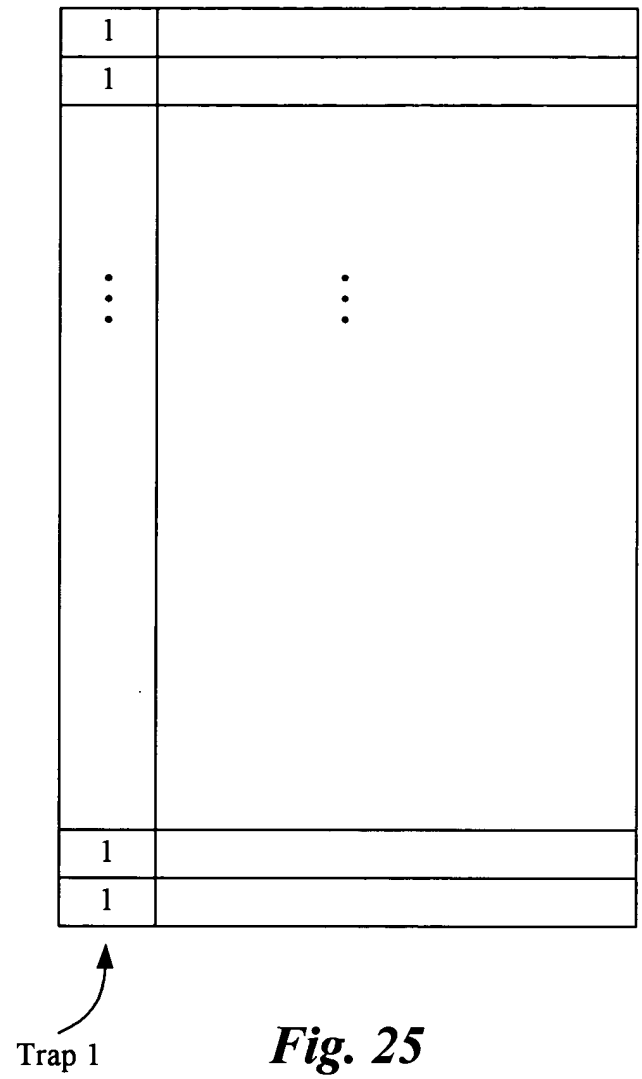

Fig. 25